P2: Building Built in Minutes: NeRF (Phase 2)

1st Venkateshkrishna
Masters in Robotics
Worcester Polytechnic Institute
Worcester, MA 01609
vparsuram@wpi.edu

2nd Mayank, Bansal
Masters in Robotics
Worcester Polytechnic Institute
Worcester, MA 01609
mbansal1@wpi.edu

I. Phase 2: Deep Learning Approach

A. Introduction

NeRF has introduced a transformative approach in the representation of 3D scenes, allowing for the creation of new viewpoints of intricate scenes by refining an underlying continuous volumetric scene function from a select few input views. This technique employs a fully connected neural network to depict a scene, where the network receives a singular continuous 5D coordinate—encompassing spatial position (x, y, z) and viewing direction (θ, ψ)—as its input. It then returns the volume density and RGB color values corresponding to that specific viewing direction as output.

B. Input Data

The official Lego and Ship dataset for NeRF, which is freely accessible on the University of Berkeley's website, is utilized. Moreover, spatial positions (x, y, z) and viewing directions (θ, ψ) are included. Despite NeRF being a deep learning methodology, it incorporates traditional techniques on the data prior to inputting it into the network.

C. Generation of Rays

In this methodology, we employ the traditional technique of volume rendering, interpreting every pixel within an image as a ray emanating into the 3D environment. Initially, the process involves converting the pixel coordinates (u, v) to normalized coordinates $(X, Y, 1)$, relative to the camera's center.

The equation for a generic ray is expressed as:

$$r(t) = o + td$$

where o denotes the ray's origin, corresponding to the position of the image pixel within the 3D space, d indicates the ray's direction (a unit vector leading from the camera's center to the image pixel), and t serves as a scalar parameter. While t is conceptually continuous, for practical implementation, it is sampled at discrete intervals.

Given that the direction of the ray is initially specified in relation to the camera's framework, we implement the rotation matrix that aligns with the camera-to-world rotation, as specified by the transformation matrix. This adjustment provides us with the direction of the ray in world coordinates, which is subsequently normalized to a unit vector.

D. Point Sampling in the ray

Points are sampled along the ray to acquire values that serve as inputs for our model. While linear sampling has been utilized, the method proves to be compatible with non-linear sampling too. Furthermore, a slight perturbation is introduced to the sampling positions, exposing the network to new data points and consequently improving the results.

E. Positional and Directional Encodings

Directly feeding the network with the coordinates of points typically leads to suboptimal performance, observable even when processing a solitary image. This issue arises due to the network's inclination towards assimilating predominantly low-frequency features while neglecting the high-frequency details. To mitigate this, positional encodings are employed, where the sines and cosines of the point coordinates, calculated across a spectrum of frequencies, are supplied to the network in lieu of the raw coordinates. In a similar vein, for directional inputs, an approach known as Directional Encodings is utilized, which incorporates the sines and cosines of the directional inputs at various frequencies.

F. Volumetric Rendering

The NeRF network produces RGB colors and volume density as outputs for specified spatial and directional inputs. These outputs are integrated using a volume rendering equation to derive the color values at particular points in world space. The equation employed is outlined as follows:

$$C \approx \sum_{i=1}^{N} T_i \alpha_i c_i$$

In this context, T_i denotes the weights, and c_i signifies the colors.

The weights T_i are calculated through:

$$T_i = \prod_{j=1}^{i-1} (1 - \alpha_j)$$

while α_i is given by:

$$\alpha_i = 1 - e^{-\sigma_i \delta t_i}$$

To determine the overall transmittance to a particular sample, we utilize the density values, which are then applied alongside the RGB colors at that location to compute the

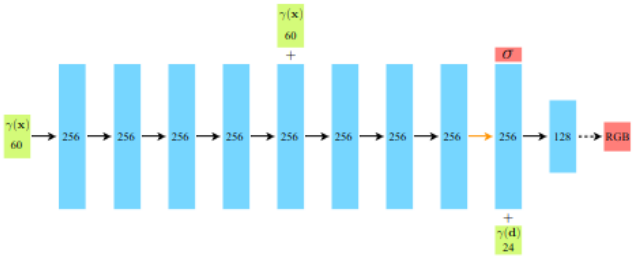

Fig. 1: NeRF Architecture

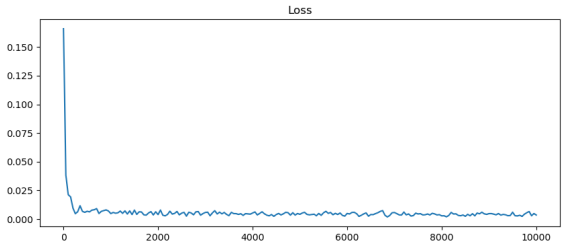

Fig. 2: Training Loss for Lego dataset

ultimate color values in the resulting image. This method is applied uniformly across all sampling points on every ray, thereby constituting what is known as the radiance field.

G. NeRF: Network

The actual network involves fully connected networks (Multi-Layer Perceptron) with the inputs being positional and directional encodings and the outputs being the density and the RGB values. The network involves skipped connection to obtain more accurate results. The network involves ReLU and Sigmoid activations. The architecture for NeRF can be seen in Fig. .

H. Network Parameters

The network parameters are:

- Input image size = 100x100
- near distance = 2
- far distance = 6
- Number of spatial samples per ray = 64
- Learning rate = 5e-4
- Number of encoding functions for samples = 10
- Loss function to be minimized: Log of mean-squared loss between predicted image (RGB values) and the actual image values

I. Results

	PSNR	SSIM
Lego with positional encoding	23.5	0.84
Lego without positional encoding	18.0	0.66
Ship with positional encoding	24.0	0.73

TABLE I: Comparison of PSNR and SSIM Values

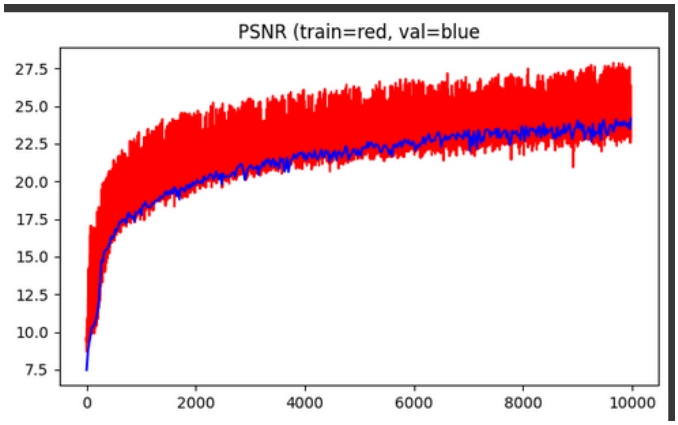

Fig. 3: PSNR for Lego dataset

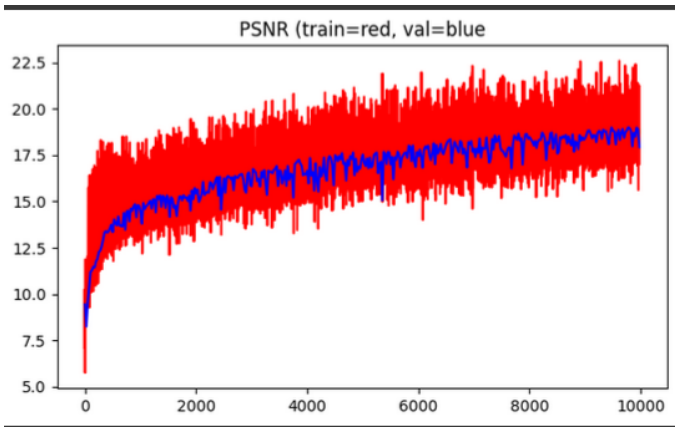

Fig. 4: PSNR for Lego dataset without positional encoding

J. Conclusion

Given the constraints of our computational resources, as well as the scaled-down network design and diminished image dimensions employed, we secured satisfactory outcomes, with the 3D model being clearly observable. While the original NeRF architecture has been successfully integrated into our codebase, these limitations necessitated the adoption of the aforementioned smaller-scale network. Employing the comprehensive network model in conjunction with the original image sizes would likely yield enhanced results that more accurately mirror the actual scenario.

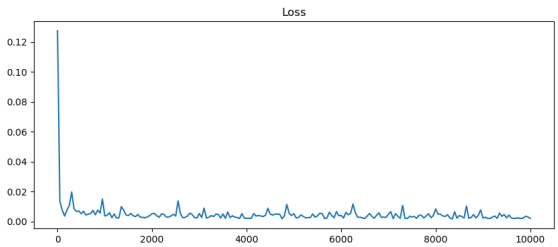

Fig. 5: Training Loss for Ship dataset

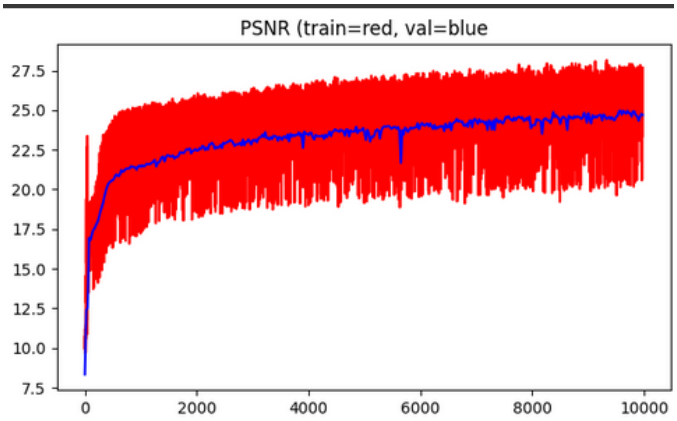

Fig. 6: PSNR for Ship dataset

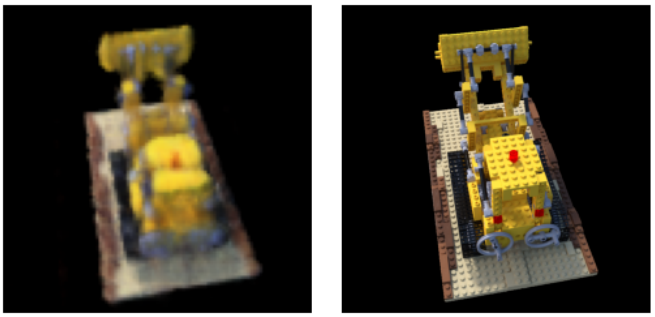

Fig. 7: Comparison of Lego generated image with positional encoding with the ground-truth test image

REFERENCES

[1] Building Build in Minutes-NeRF: link

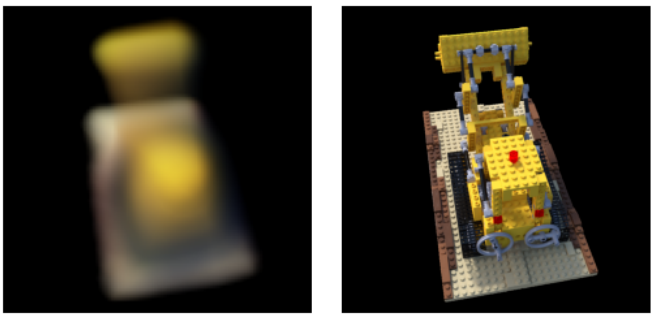

Fig. 8: Comparison of Lego generated image without positional encoding with the ground-truth test image

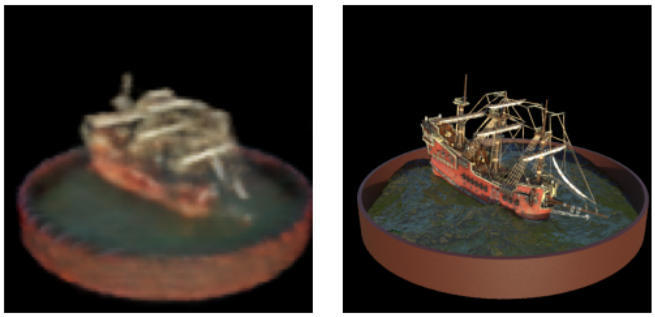

Fig. 9: Comparison of Ship generated image with the ground-truth test image

Fig. 10: Novel synthesised view of Lego with positional encoding

Fig. 11: Novel synthesised view of Lego without positional encoding

Fig. 12: Novel synthesised view of Ship without positional encoding